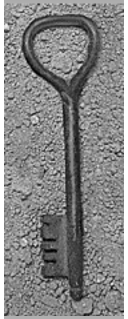
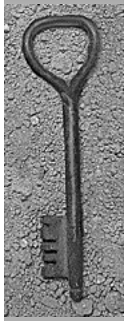
Key Establishment Schemes Workshop Document

October 2001



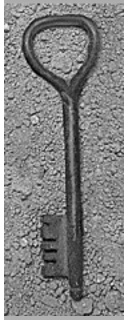
Outline

- Introduction
- Scope & Purpose
- Definitions
- Key Establishment Algorithm Classes
- Security Attributes
- Cryptographic Elements
- Key Agreement Schemes
- Key Transport
- Keys Derived from a “Master Key”
- Key Recovery



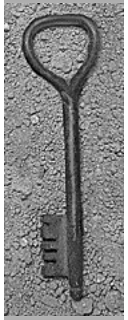
Introduction

- ♦ Many cryptographic algorithms (e.g., AES, HMAC) require the establishment of *shared keying material* in advance.
- ♦ Manual distribution of keying material is inefficient and complex.
- ♦ Seek automated key establishment schemes.



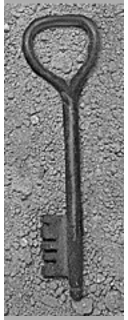
Scope & Purpose

- ♦ Development of a Federal key agreement schemes document based on
  - – **ANSI X9.42** Agreement of Symmetric Keys using Discrete Logarithm Cryptography
  - – **ANSI X9.44** Key Agreement and Key Transport using Factoring-Based Cryptography (To be provided)
  - – **ANSI X9.63** Key Agreement and Key Transport using Elliptic Curve Cryptography



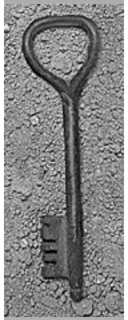
Definitions

- Approved
  - FIPS approved or NIST Recommended
- Keying Material
  - The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.
- Shared Keying Material
  - The keying material that is derived by applying a key derivation function to the shared secret.
- Shared Secret
  - A secret value computed using a prescribed algorithm and combination of keys belonging to the participants in the key establishment scheme.



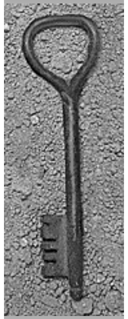
General Symbols

H	An approved hash function
[$Text_1$], [$Text_2$]	An optional bit string that may be used during key confirmation and that is sent between the parties establishing keying material
U	One entity of a key establishment process, or the bit string denoting the identity of that entity
V	The other entity of a key establishment process, or the bit string denoting the identity of that entity
$X\|\|Y$	Concatenation of two strings X and Y



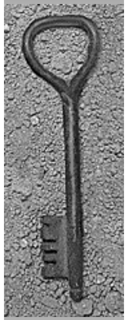
ANSI X9.42 Symbols

p, q, g	The domain parameters
mod *p*	The reduction modulo *p* on an integer value
$r_{U,}$ r_V	Party U or Party V's ephemeral private key
$t_{U,}$ t_V	Party U or Party V's ephemeral public key
$x_{U,}$ x_V	Party U or Party V's static private key
$y_{U,}$ y_V	Party U or Party V's static public key
Z	A shared secret that is used to derive keying material using a key derivation function
Z_e	An ephemeral shared secret that is computed using the Diffie-Hellman primitive
Z_s	A static shared secret that is computed using the Diffie-Hellman primitive



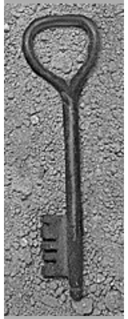
ANSI X9.63 Symbols

[*X*]	Indicates that the inclusion of the bit string or octet string *X* is optional
a, *b*	Field elements that define the equation of an elliptic curve
avf(*P*)	The associate value of the elliptic curve point
$d_{e,U}$, $d_{e,V}$	Party U's and Party V's ephemeral private keys
$d_{s,U}$, $d_{s,V}$	Party U's and Party V's static private keys
FR	An indication of the basis used
G	A distinguished point on an elliptic curve
h	The cofactor of the elliptic curve



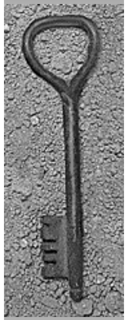
ANSI X9.63 Symbols

n	The order of the point G
q	The field size
j	A special point on an elliptic curve, called the point at infinity. The additive identity of the elliptic curve group.
$Q_{e,U}$, $Q_{e,V}$	Party U's and Party V's ephemeral public keys
$Q_{s,U}$, $Q_{s,V}$	Party U's and Party V's static public keys
SEED	An optional bit string that is present if the elliptic curve was randomly generated
x_P	The x-coordinate of a point P.
y_P	The y-coordinate of a point P.
Z	A shared secret that is used to derive key using a key derivation function
Ze	An ephemeral shared secret that is computed using the Diffie-Hellman primitive
Zs	A static shared secret that is computed using the Diffie-Hellman primitive



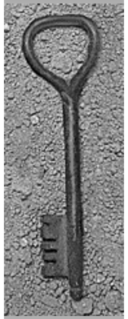
Key Establishment Algorithm Classes

- ♦ Cryptographic keying material may be electronically established between parties using either key agreement or key transport schemes.
- ♦ During key agreement, the keying material to be established is not sent; information is exchanged between the parties that allow the calculation of the keying material. Key agreement schemes use asymmetric (public key) techniques.
- ♦ During key transport, encrypted keying material is sent from an initiator who generates the keying material to another party. Key transport schemes use either symmetric or public key techniques.



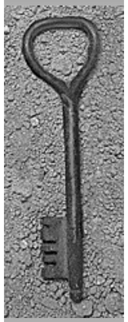
Security Attributes

- ♦ To be determined…



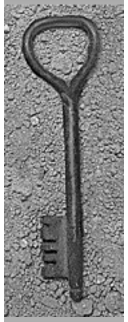
Cryptographic Elements

- ♦ Domain Parameters (Generation, Validation, and Management)
- ♦ Private/Public Keys (Generation, PK Validation, Management)
- ♦ Key Derivation Function
- ♦ Message Authentication Code
- ♦ Associate Value Function (Elliptic Curves Only)
- ♦ Cryptographic Hash Functions
- ♦ Random Number Generation
- ♦ Key Confirmation
- ♦ Calculation of Shared Secrets
- ♦ RSA Primitives (To be provided)
- ♦ Key Wrapping Primitive(s) (To be provided)



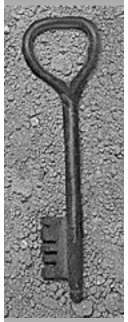
Domain Parameter Generation

- ♦ ANSI X9.42 Requirements
  - – (p,q,g) where p and q are prime, and g is the generator of the q-order cyclic subgroup of GF(p)
- ♦ ANSI X9.63 Requirements
  - – (q, FR, a, b, [SEED], G, n, h) where q (field size), FR (basis used), a and b (field elements), SEED (optional bit string), G (point), n (order of the point G), and h (cofactor).



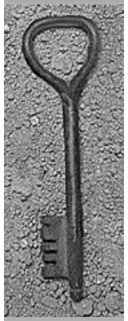
Domain Parameter Validation

- ♦ One of three methods must be employed before use
  - – The party generates (and checks) the parameters
  - – The party validates parameters as specified in appropriate ANSI standards
  - – The party receives assurance from a trusted party (e.g., a CA) that the parameters are valid by one of the above methods



Domain Parameter Management

- ♦ Only authorized (trusted) parties should generate domain parameters
- ♦ Key pairs must be associated with their domain parameters
- ♦ Modification or substitution of domain parameters may cause security risks



Private/Public Keys

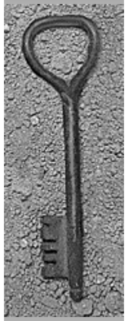
- ♦ Key Pair Generation
  - – Static and ephemeral key pairs are generated using the same primitives
  - – Private keys must be created using an approved RNG
- ♦ Public Key Validation
  - – Static public keys **must** be validated by the recipient, or by an entity that is trusted by the recipient
  - – Each ephemeral public key **must** be validated by the recipient before being used to derive a shared secret
- ♦ Key Pair Management
  - – Public/private key pairs **must** be correctly associated with their corresponding domain parameters
  - – Static public keys **must** be obtained in a trusted manner
  - – Ephemeral keys **must** be destroyed immediately after the shared secret is computed

Cryptographic Elements

- ♦ Key Derivation Function (KDF)
  - – Used to derive keying material from a shared secret
  - – Uses identities of communicating parties
- ♦ Message Authentication Code (MAC)
  - – A function of both a symmetric key and data
  - – MAC function used to provide key confirmation
- ♦ Associate Value Function (EC Only)
  - – Used by the MQV family of key agreement schemes to compute an integer associated with an elliptic curve point

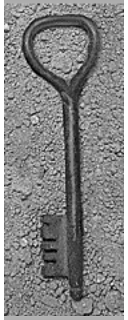
Cryptographic Elements

- ♦ Cryptographic Hash Functions
  - – Use approved hash functions whenever required.
- ♦ Random Number Generation
  - – Use approved random number generators whenever required
- ♦ Key Confirmation
  - – Used to provide assurance that the parties have derived the same keys



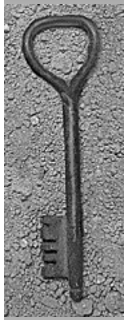
Calculation of Shared Secrets

- ♦ Use DH of ANSI X9.42 for dhHybrid1, dhEphem, dhHybridOneFlow, dhOneFlow, and dhStatic schemes
- ♦ Use Modified DH of ANSI X9.63 for Full Unified Model, Ephemeral Unified Model, 1-Pass Unified Model, 1-Pass Diffie-Hellman, and Static Unified Model Schemes (Differs from ANSI X9.63)



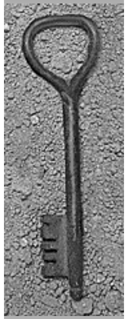
Calculation of Shared Secrets

- ♦ Use MQV2 primitive of ANSI X9.42 for the MQV2 scheme
- ♦ Use MQV1 primitive of ANSI X9.42 for MQV1 scheme
- ♦ Use MQV primitive of Section 5.5 of ANSI X9.63 for Full MQV and 1-Pass MQV schemes
- ♦ Shared Secrets
  - – **must not** be used directly as shared keying material.
  - – **must** be calculated by applying a key derivation function to the shared secret.



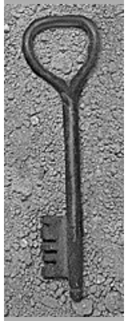
Other Primitives

- ♦ RSA Primitives
  - – To be addressed later…
- ♦ Key Wrapping Primitive(s)
  - – To be addressed later…



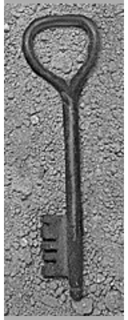
Key Agreement Schemes Categories

- ♦ **C(2): Two Party Participation**
  - – *Interactive, 2-way*
  - – Each party generates an ephemeral key pair.
- ♦ **C(1): One Party Participation**
  - – *Store-and-Forward, 1-way*
  - – Only the initiator generates an ephemeral key pair.
- ♦ **C(0): Static Keys Only**
  - – *Static (passive)*
  - – No ephemeral keys are used.



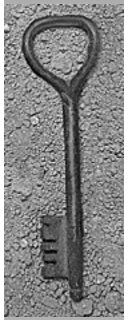
Key Agreement Schemes Subcategories

- C(2,2): Each party generates an ephemeral key pair and has a static key pair.
- C(2,0): Each party generates an ephemeral key pair; no static keys are used.
- C(1,2): The initiator generates an ephemeral key pair and has a static key pair; the responder has a static key pair.
- C(1,1): The initiator generates an ephemeral key pair, but has no static key pair; the responder has only a static key pair.
- C(0,2): Each party has only static keys.



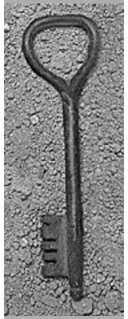
Key Agreement Schemes Subcategories

- Primitive: Either a DH or an MQV primitive
- Arithmetic: Either FF as in ANSI X9.42 or EC as in ANSI X9.63
- Example: dhHybrid1 can be classified as C(2, 2, DH, FF)



Key Agreement Schemes

Category	**Subcategory**	**Primitive**	**Arith.**	**Scheme**	**Full Classification**
C(2)	*C(2,2)*	DH	FF	dhHybrid1	C(2,2,DH,FF)
C(2)	*C(2,2)*	DH	EC	Full Unified Model	C(2,2,DH,EC)
C(2)	*C(2,2)*	MQV	FF	MQV2	C(2,2,MQV,FF)
C(2)	*C(2,2)*	MQV	EC	Full MQV	C(2,2,MQV,EC)
C(2)	*C(2,0)*	DH	FF	dhEphem	C(2,0,DH,FF)
C(2)	*C(2,0)*	DH	EC	Ephemeral Unified Model	C(2,0,DH,EC)
C(1)	*C(1,2)*	DH	FF	dhHybridOneFlow	C(1,2,DH,FF)
C(1)	*C(1,2)*	DH	EC	1-Pass Unified Model	C(1,2,DH,EC)
C(1)	*C(1,2)*	MQV	FF	MQV1	C(1,2,MQV,FF)
C(1)	*C(1,2)*	MQV	EC	1-Pass MQV	C(1,2,MQV,EC)
C(1)	*C(1,1)*	DH	FF	dhOneFlow	C(1,1,DH,FF)
C(1)	*C(1,1)*	DH	EC	1-Pass Diffie-Hellman	C(1,1,DH,EC)
C(0)	*C(0,2)*	DH	FF	dhStatic	C(0,2,DH,FF)
C(0)	*C(0,2)*	DH	EC	Static Unified Model	C(0,2,DH,EC)



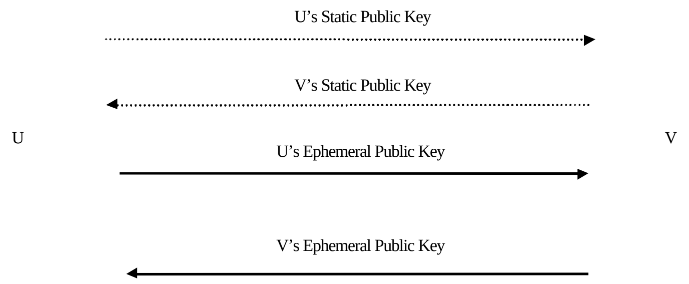
Key Agreement Schemes Overview

- ♦ Each party in a key agreement process **must** use the same domain parameters.
- ♦ These parameters **must** be established prior to the initiation of the key agreement process.
- ♦ Static public keys may be obtained from other entity or trusted third party (e.g., a CA)

Two Party Participation C(2)

- ♦ Each party generates an ephemeral key pair and has a static key pair
- ♦ *Four* C(2,2) schemes
  - – dhHybrid1
  - – Full Unified Model
  - – MQV2
  - – Full MQV

Figure 1: General Protocol when each party has both static and ephemeral key pairs



1. U uses its static and ephemeral private keys and V's static and ephemeral public keys to compute a shared secret.
2. U invokes the Key Derivation Function using the shared secret.

1. V uses its static and ephemeral private keys and U's static and ephemeral public keys to compute a shared secret.
2. V invokes the Key Derivation Function using the shared secret.

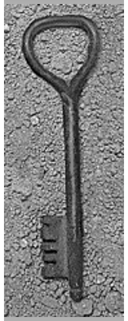


Table 4: dhHybrid1 Key Agreement Scheme C(2,2,DH,FF)

	Party U	Party V
Static Data	1. Static private key x_U 2. Static public key y_U	1. Static private key x_V 2. Static public key y_V
Ephemeral Data	1. Ephemeral private key r_U 2. Ephemeral public key t_U	1. Ephemeral private key r_V 2. Ephemeral public key t_V
Input	$(p, q, g), x_U, y_V, r_U, t_V$	$(p, q, g), x_V, y_U, r_V, t_U$
Computation	$Z_s = y_V^{x_U} \bmod p$ $Z_e = t_V^{r_U} \bmod p$	$Z_s = y_U^{x_V} \bmod p$ $Z_e = t_U^{r_V} \bmod p$
Derive Key Material	Compute *kdf*(*Z,OtherInput*) using $Z = Z_e \mid\mid Z_s$	Compute *kdf*(*Z,OtherInput*) using $Z = Z_e \mid\mid Z_s$

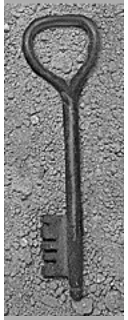


Table 5: Full Unified Model Key Agreement Scheme C(2,2,DH,EC)

	Party U	Party V
Static Data	1. Static private key $d_{s,U}$ 2. Static public key $Q_{s,U}$	1. Static private key $d_{s,V}$ 2. Static public key $Q_{s,V}$
Ephemeral Data	1. Ephemeral private key $d_{e,U}$ 2. Ephemeral public key $Q_{e,U}$	1. Ephemeral private key $d_{e,V}$ 2. Ephemeral public key $Q_{e,V}$
Input	(*q, FR a, b, [SEED], G, n, h*), $d_{e,U}, Q_{e,V}, d_{s,U}, Q_{s,V}$	(*q, FR, a, b, [SEED] G, n, h*), $d_{e,V}, Q_{e,U}, d_{s,V}, Q_{s,U}$
Computation	$(x_s, y_s) = hd_{s,U}Q_{s,V}$ $(x_e, y_e) = hd_{e,U}Q_{e,V}$ $Z_s = x_s$ $Z_e = x_e$	$(x_s, y_s) = hd_{s,V}Q_{s,U}$ $(x_e, y_e) = hd_{e,V}Q_{e,U}$ $Z_s = x_s$ $Z_e = x_e$
Derive Keying Material	Compute *kdf*(*Z,OtherInput*) using $Z= Z_e \mid\mid Z_s$	Compute *kdf*(*Z,OtherInput*) using $Z= Z_e \mid\mid Z_s$

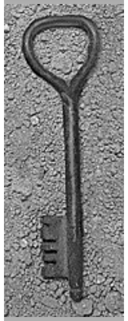


Table 6: MQV2 Key Agreement Scheme C(2,2,MQV,FF)

	Party U	Party V
Static Data	1. Static private key x_U 2. Static public key y_U	1. Static private key x_V 2. Static public key y_V
Ephemeral Data	1. Ephemeral private key r_U 2. Ephemeral public key t_U	1. Ephemeral private key r_V 2. Ephemeral public key t_V
Input	(p, q, g), x_U, y_V, r_U, t_U, t_V	(p, q, g), x_V, y_U, r_V, t_V, t_U
Computation	1. $w = \lceil \|\|q\|\|/2 \rceil$ 2. $t_U\text{¢} = (t_U \bmod 2^w) + 2^w$ 3. $S_U = (r_U + t_U\text{¢}\, x_U) \bmod q$ 4. $t_V\text{¢} = (t_V \bmod 2^w) + 2^w$ 5. $Z_{MQV} = \left(t_V\, y_V^{t_V\,'}\right)^{S_U} \bmod p$	1. $w = \lceil \|\|q\|\|/2 \rceil$ 2. $t_V\text{¢} = (t_V \bmod 2^w) + 2^w$ 3. S_V $Z_{MQV} = \left(t_U\, y_U^{t\,'}\right) \bmod p.$
Derive Keying Material	Compute $kdf(Z,OtherInput)$ using $Z = Z_{MQV}$	Compute $kdf(Z,OtherInput)$ using $Z = Z_{MQV}$

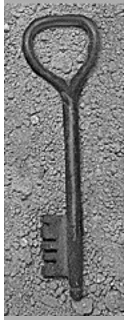
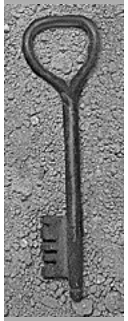


Table 7: Full MQV Key Agreement Scheme C(2,2,MQV,EC)

	Party U	Party V
Static Data	1. Static private key $d_{s,U}$ 2. Static public key $Q_{s,U}$	1. Static private key $d_{s,V}$ 2. Static public key $Q_{s,V}$
Ephemeral Data	1. Ephemeral private key $d_{e,U}$ 2. Ephemeral public key $Q_{e,U}$	1. Ephemeral private key $d_{e,V}$ 2. Ephemeral public key $Q_{e,V}$
Input	$(q, FR\ a, b, [SEED], G, n, h)$, $d_{e,U}$, $Q_{e,V}$, $d_{s,U}$, $Q_{e,U}$, $Q_{s,V}$	$(q, FR, a, b, [SEED]\ G, n, h)$, $d_{e,V}$, $Q_{e,U}$, $d_{s,V}$, $Q_{e,V}$, $Q_{s,U}$
Computation	1. $implicitsig_U = (d_{e,U} + avf(Q_{e,U})d_{s,U}) \bmod n$ 2. $(x, y) = h \times implicitsig_U \times (Q_{e,V} + avf(Q_{e,V})Q_{s,V})$ 3. $Z = x$	1. $implicitsig_V = (d_{e,V} + avf(Q_{e,V})d_{s,V}) \bmod n$ 2. $(x, y) = h \times implicitsig_V \times (Q_{e,U} + avf(Q_{e,U})Q_{s,U})$ 3. $Z = x$
Derive Keying Material	Compute $kdf(Z,OtherInput)$ using $Z = x$	Compute $kdf(Z,OtherInput)$ using $Z = x$



Two Party Participation

- Each party generates an ephemeral key pair; no static keys are used.
- *Two* C(2,0) schemes
  - dhEphem
  - Ephemeral Unified Model

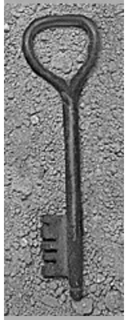
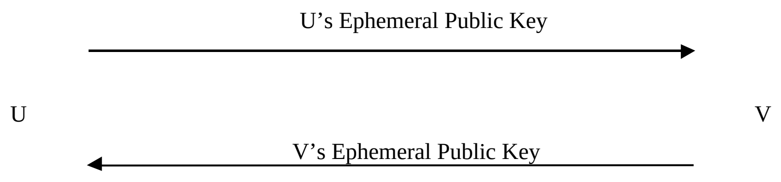


Figure 2: General protocol when each party generates ephemeral key pairs; no static keys are used



1. U uses its ephemeral private key and V's ephemeral public key to form a shared secret.
2. U invokes the Key Derivation Function using the shared secret.

1. V uses its ephemeral private key and U's ephemeral public key to form a shared secret.
2. V invokes the Key Derivation Function using the shared secret.

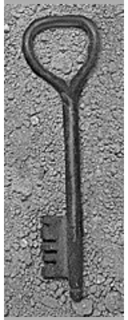


Table 8: dhEphem Key Agreement Scheme C(2,0,DH,FF)

	Party U	**Party V**
Static Data	N/A	N/A
Ephemeral Data	1. Ephemeral private key r_U 2. Ephemeral public key t_U	1. Ephemeral private key r_V 2. Ephemeral public key t_V
Input	$(p, q, g), r_U, t_V$	$(p, q, g), r_V, t_U$
Computation	$Z_e = t_V^{r_U} \bmod p$	$Z_e = t_U^{r_V} \bmod p$
Derive Keying Material	Compute *kdf(Z,OtherInput)* using $Z = Z_e$	Compute *kdf(Z,OtherInput)* using $Z = Z_e$

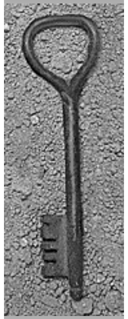


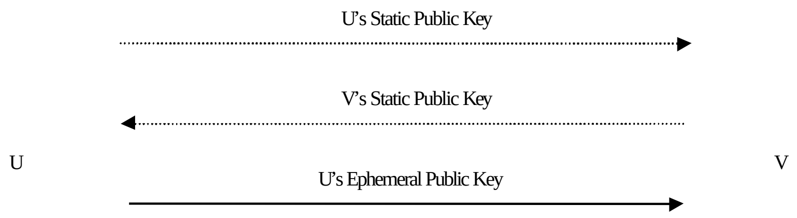
Table 9: Ephemeral Unified Model Key Agreement Scheme C(2,0,DH,EC)

	Party U	**Party V**
Static Data	N/A	N/A
Ephemeral Data	1. Ephemeral private key $d_{e,U}$ 2. Ephemeral public key $Q_{e,U}$	1. Ephemeral private key $d_{e,V}$ 2. Ephemeral public key $Q_{e,V}$
Input	$(q, FR\ a, b, [SEED], G, n, h)$, $d_{e,U}, Q_{e,V}$	$(q, FR, a, b, [SEED]\ G, n, h)$, $d_{e,V}, Q_{e,U}$
Computation	$(x_e, y_e) = hd_{e,U}Q_{e,V}$ $Z_e = x_e$	$(x_e, y_e) = hd_{e,V}Q_{e,U}$ $Z_e = x_e$
Derive Keying Material	Compute *kdf(Z,OtherInput)* using $Z = Z_e$	Compute *kdf(Z,OtherInput)* using $Z = Z_e$

One Party Participation

- ♦ Initiator has a static key pair and generates an ephemeral key pair; Responder has a static key pair.
- ♦ *Four* C(1,2) schemes
  - – dhHybridOneFlow
  - – 1-Pass Unified Model
  - – MQV1
  - – 1-Pass MVQ

Figure 3: General protocol when the Initiator has both static and ephemeral key pairs, and the Responder has only a static key pair



1. U uses its static and ephemeral private keys and V’s static public key to form a shared secret
2. U invokes the Key Derivation Function using the shared secret

1. V uses its static private key and U’s static and ephemeral public key to form a shared secret
2. V invokes the Key Derivation Function using the shared secret

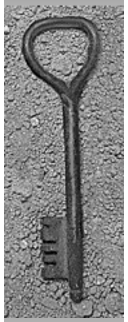


Table 10: dhHybridOneFlow Key Agreement Scheme C(1,2,DH,FF)

	Party U	**Party V**
Static Data	1. Static private key x_U 2. Static public key y_U	1. Static private key x_V 2. Static public key y_V
Ephemeral Data	1. Ephemeral private key r_U 2. Ephemeral public key t_U	N/A
Input	(p, q, g), x_U, r_U, y_V	(p, q, g), x_V, y_U, t_U
Computation	$Z_s = y_V^{x_U} \bmod p$ $Z_e = y_V^{r_U} \bmod p$	$Z_s = y_U^{x_V} \bmod p$ $Z_e = t_U^{x_V} \bmod p$
Derive Keying Material	Compute *kdf*(*Z*,*OtherInput)* using $Z = Z_e \mid\mid Z_s$	Compute *kdf*(*Z*,*OtherInput)* using $Z = Z_e \mid\mid Z_s$

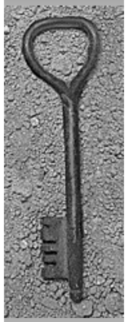


Table 11: 1-Pass Unified Model Key Agreement Scheme C(1,2,DH,EC)

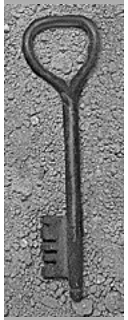
	Party U	**Party V**
Static Data	1. Static private key $d_{s,U}$ 2. Static public key $Q_{s,U}$	1. Static private key $d_{s,V}$ 2. Static public key $Q_{s,V}$
Ephemeral Data	1. Ephemeral private key $d_{e,U}$ 2. Ephemeral public key $Q_{e,U}$	N/A
Input	$(q, FR, a, b, [SEED], G, n, h)$, $d_{s,U}$, $d_{e,U}$, $Q_{s,V}$	$(q, FR, a, b, [SEED], G, n, h)$, $d_{s,V}$, $Q_{s,U}$, $Q_{e,U}$
Computation	$(x_s, y_s) = h\, d_{s,U}\, Q_{s,V}$ $(x_e, y_e) = h\, d_{e,U}\, Q_{s,V}$ $Z_s = x_s$ $Z_e = x_e$	$(x_s, y_s) = h\, d_{s,V}\, Q_{s,U}$ $(x_e, y_e) = h\, d_{s,V}\, Q_{e,U}$ $Z_s = x_s$ $Z_e = x_e$
Derive Keying Material	Compute *kdf*(*Z*,*OtherInput)* using $Z = Z_e \mid\mid Z_s$	Compute *kdf*(*Z*,*OtherInput)* using $Z = Z_e \mid\mid Z_s$

Table 12: MQV1 Key Agreement Scheme C(1,2,MQV,FF)

	Party U	Party V
Static Data	1. Static private key x_U 2. Static public key y_U	1. Static private key x_V 2. Static public key y_V
Ephemeral Data	1. Ephemeral private key r_U 2. Ephemeral public key t_U	N/A
Input	$(p, q, g), x_U, y_V, r_U, t_U$	$(p, q, g), x_V, y_U, t_U$
Computation	1. $w = \lceil \lVert q \rVert / 2 \rceil$ 2. $t_U' = (t_U \bmod 2^w) + 2^w$ 3. $S_U = (r_U + t_U' x_U) \bmod q$ 4. $y_V' = (y_V \bmod 2^w) + 2^w$ 5. $Z_{MQV} = \left(y_V \, y_V^{y_V'}\right)^{S_U} \bmod p$	1. $w = \lceil \lVert q \rVert / 2 \rceil$ 2. y_V¢ $= (y_V \bmod 2^w) + 2^w$ 3. $S_V = (x_V + y_V$¢$x_V) \bmod q$ 4. $t_U' = (t_U \bmod 2^w) + 2^w$ 5. $Z_{MQV} = \left(t_U \, y_U^{t_U'}\right)^{S_V} \bmod p$
Derive Keying Material	Compute *kdf(Z,OtherInput)* using $Z = Z_{MQV}$	Compute *kdf(Z,OtherInput)* using $Z = Z_{MQV}$

Table 13:1-Pass MQV Model Key Agreement Scheme C(1,2,MQV,EC)

	Party U	Party V
Static Data	1. Static private key $d_{s,U}$ 2. Static public key $Q_{s,U}$	1. Static private key $d_{s,V}$ 2. Static public key $Q_{s,V}$
Ephemeral Data	1. Ephemeral private key $d_{e,U}$ 2. Ephemeral public key $Q_{e,U}$	N/A
Input	$(q, FR, a, b, [SEED], G, n, h), d_{e,U}, d_{s,U}, Q_{e,U}, Q_{s,V}$	$(q, FR, a, b, [SEED], G, n, h), d_{s,V}, Q_{s,V}, Q_{e,U}, Q_{s,U}$
Computation	1. $implicitsig_U = (d_{e,U} + avf(Q_{e,U})d_{s,U}) \bmod n$ 2. $(x, y) = h \times implicitsig_U \times (Q_{s,V} + avf(Q_{s,V})\, Q_{s,V})$ 3. $Z = x$	*1.* $implicitsig_V = (d_{s,V} + avf(Q_{s,V})d_{s,V}) \bmod n$ *2.* $(x, y) = h \times implicitsig_V \times (Q_{e,U} + avf(Q_{e,U})\, Q_{s,U})$ *3.* $Z = x$
Derive Keying Material	Compute *kdf(Z,OtherInput)* using $Z = x$	Compute *kdf(Z,OtherInput)* using $Z = x$



One Party Participation

- ♦ Initiator generates only an ephemeral key pair; Responder has only a static key pair.
- ♦ *Two* C(1,1) schemes
  - – dhOneFlow
  - – 1-Pass Diffie-Hellman

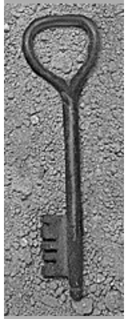
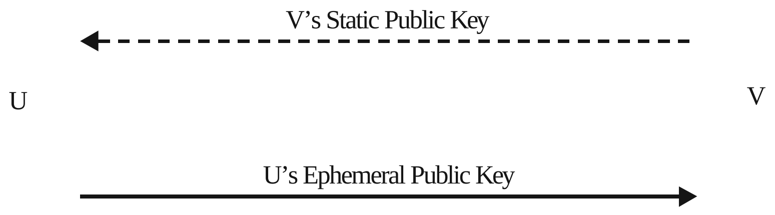


Figure 4: General protocol when the Initiator has only an ephemeral key pair, and the Responder has only a static key pair



1. U uses its ephemeral private key and V's static public key to form a shared secret
2. U invokes the Key Derivation Function using the shared secret

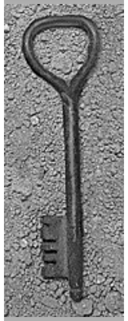
1. V uses its static private key and U's ephemeral public key to form a shared secret
2. V invokes the Key Derivation Function using the shared secret

Table 14: dhOneFlow Key Agreement Scheme C(1,1,DH,FF)

	Party U	Party V
Static Data	N/A	1. Static private key x_V 2. Static public key y_V
Ephemeral Data	1. Ephemeral private key r_U 2. Ephemeral public key t_U	N/A
Input	$(p, q, g), r_U, y_V$	$(p, q, g), x_V, t_U$
Computation	$Z_e = y_V^{r_U} \bmod p$	$Z_e = t_U^{x_V} \bmod p$
Derive Keying Material	Compute *kdf(Z,OtherInput)* using $Z = Z_e$	Compute *kdf(Z,OtherInput)* using $Z = Z_e$

Table 15: 1-Pass Diffie-Hellman Model Key Agreement Scheme C(1,1,DH,EC)

	Party U	Party V
Static Data	N/A	1. Static private key $d_{s,V}$ 2. Static public key $Q_{s,V}$
Ephemeral Data	1. Ephemeral private key $d_{e,U}$ 2. Ephemeral public key $Q_{e,U}$	N/A
Input	$(q, FR, a, b, [SEED], G, n, h), d_{e,U}, Q_{s,V}$	$(q, FR, a, b, [SEED], G, n, h), d_{s,V}, Q_{e,U}$
Computation	$(x, y) = h\, d_{e,U} Q_{s,V}$ $Z = x$	$(x, y) = h\, d_{s,V} Q_{e,U}$ $Z = x$
Derive Keying Material	Compute *kdf*(*Z*,*OtherInput*) using $Z = x$	Compute *kdf*(*Z*,*OtherInput*) using $Z = x$



Static Keys Only

- ♦ Each party has only a static key pair
- ♦ Two C(0,2) schemes
  - – dhStatic
  - – Static Unified Model

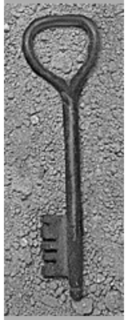
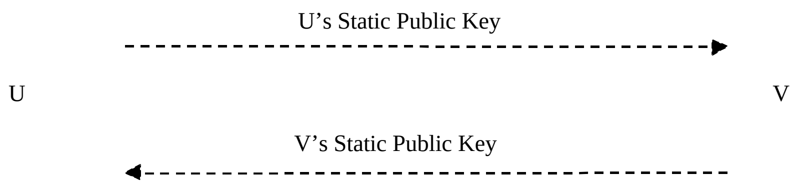


Figure 5: Each party has only a static key pair



1. U uses its static private key and V's static public key to form a shared secret
2. U invokes the Key Derivation Function using the shared secret

1. V uses its static private key and U's static public key to form a shared secret
2. U invokes the Key Derivation Function using the shared secret

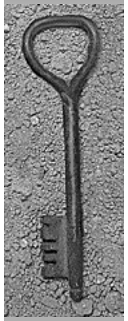


Table 16: dhStatic Key Agreement Scheme C(0,2,DH,FF)

	Party U	Party V
Static Data	1. Static private key x_U 2. Static public key y_U	1. Static private key x_V 2. Static public key y_V
Ephemeral Data	N/A	N/A
Input	$(p, q, g), x_U, y_V$	$(p, q, g), x_V, y_U$
Computation	$Z_s = y_V^{x_U} \bmod p$	$Z_s = y_U^{x_V} \bmod p$
Derive Keying Material	Compute *kdf(Z,OtherInput)* using Z=Z$_s$	Compute *kdf(Z,OtherInput)* using Z=Z_s

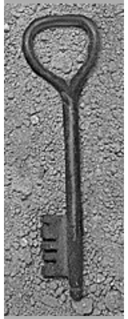


Table 17: Static Unified Model Key Agreement Scheme C(0,2,DH,EC)

	Party U	Party V
Static Data	1. Static private key $d_{s,U}$ 2. Static public key $Q_{s,U}$	1. Static private key $d_{s,V}$ 2. Static public key $Q_{s,V}$
Ephemeral Data	N/A	N/A
Input	$(q, FR, a, b, [SEED], G, n, h), d_{s,U}, Q_{s,V}$	$(q, FR, a, b, [SEED], G, n, h), d_{s,V}, Q_{s,U}$
Computation	$(x_s, y_s) = hd_{s,U}Q_{s,V}$ $Z_s = x_s$	$(x_s, y_s) = hd_{s,V}Q_{s,U}$ $Z_s = x_s$
Derive Keying Material	Compute *kdf(Z,OtherInput)* using $Z = Z_s$	Compute *kdf(Z,OtherInput)* using $Z = Z_s$

Topics to be Addressed

- ♦ Key Transport
  - – To be addressed
- ♦ Keys Derived from a “Master Key”
  - – Suggestions welcome

Key Recovery

- ♦ Some applications may desire to recover protected data by first recovering the associated key
- ♦ Static key pairs may be saved (See Key Management Guideline document)
- ♦ Static public keys may be saved (e.g., public key certificates)
- ♦ Ephemeral public keys may be saved
- ♦ Ephemeral private keys **must not** be recoverable or saved

Implementation Validation

- ♦ Implementations of schemes in the final schemes document must be tested in order to claim compliance
- ♦ For information on NIST’s testing program see http://csrc.nist.gov/cryptval

Questions?

Give me a break!

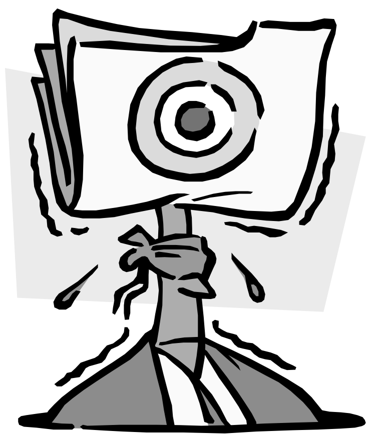
Discussion Topics

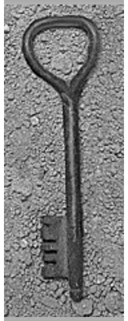
- ♦ Are there any situations which are not addressed by at least one of the schemes in the document?
- ♦ Which schemes should use key confirmation?
- ♦ Should key confirmation ever be mandatory?
- ♦ Does it unnecessarily hinder any application to require a distinction between initiator and responder in a scheme?
- ♦ Should the identities of the initiator and responder be used in the calculation of shared secrets? (related to previous question)

Discussion Topics

- ♦ Should this document address broader forms of key derivation (e.g., key derivation for multi-user applications)?
- ♦ What are the most important key establishment scheme attributes, and how should they be presented? (Please bring your ideas)
- ♦ Are there any additional topics that should be covered?
- ♦ Are there any additional appendices that should be included?

Questions or Discussion?





Closing

- ♦ Thanks for coming and helping
- ♦ See http://www.nist.gov/kms
- ♦ We will let you know when report is posted
- ♦ Send comments to kmscomments@nist.gov
- ♦ Have a safe trip home